Text-to-Image Generation with Diffusion Models

Lecture 11

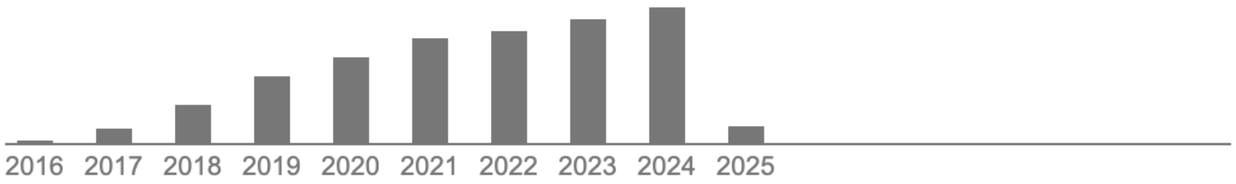
18-789

Citation Trend: Diffusion vs GAN

First GAN paper

Generative adversarial networks

Authors	Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio
Publication date	2014

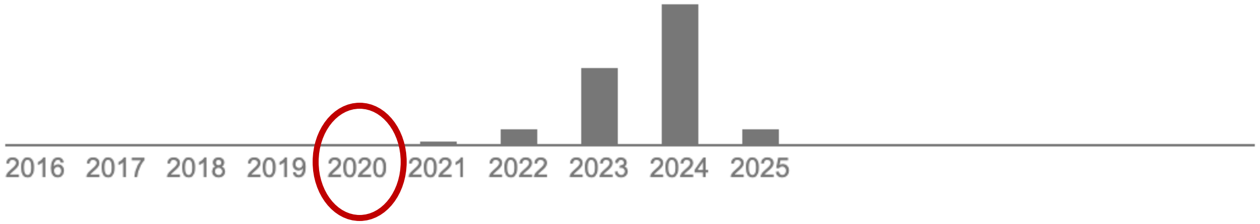


What happened around 2020?

First Diffusion paper

Deep unsupervised learning using nonequilibrium thermodynamics

Authors	Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, Surya Ganguli
Publication date	2015/3/12



Outline: Text-to-Image Diffusion Models

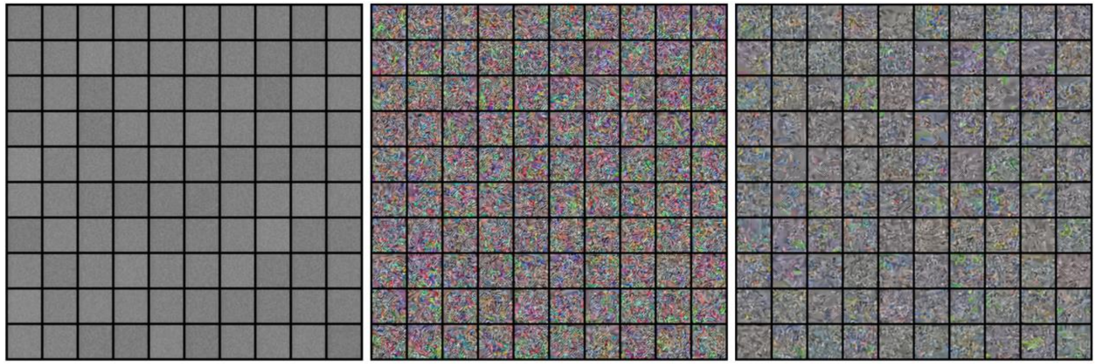
- Network architecture
  - Noise-level Conditioning
  - Latent-space Modeling
  - U-Net vs. Transformers
- Classifier-free Guidance (CFG)

Noise-level Conditional Diffusion Model

- It’s crucial to train a single diffusion network on all time steps (and condition it on the time step).

$$\|s_\theta(x; t) - \nabla_x \log p_t(x)\|^2$$

Time-step conditioned



NOT time-step conditioned

Yang and Ermon, “Generative Modeling by Estimating Gradients of the Data Distribution”, NeurIPS 2019

Noise-level Conditional Diffusion Model

- It's crucial to train a single diffusion network on all time steps (and condition it on the time step).

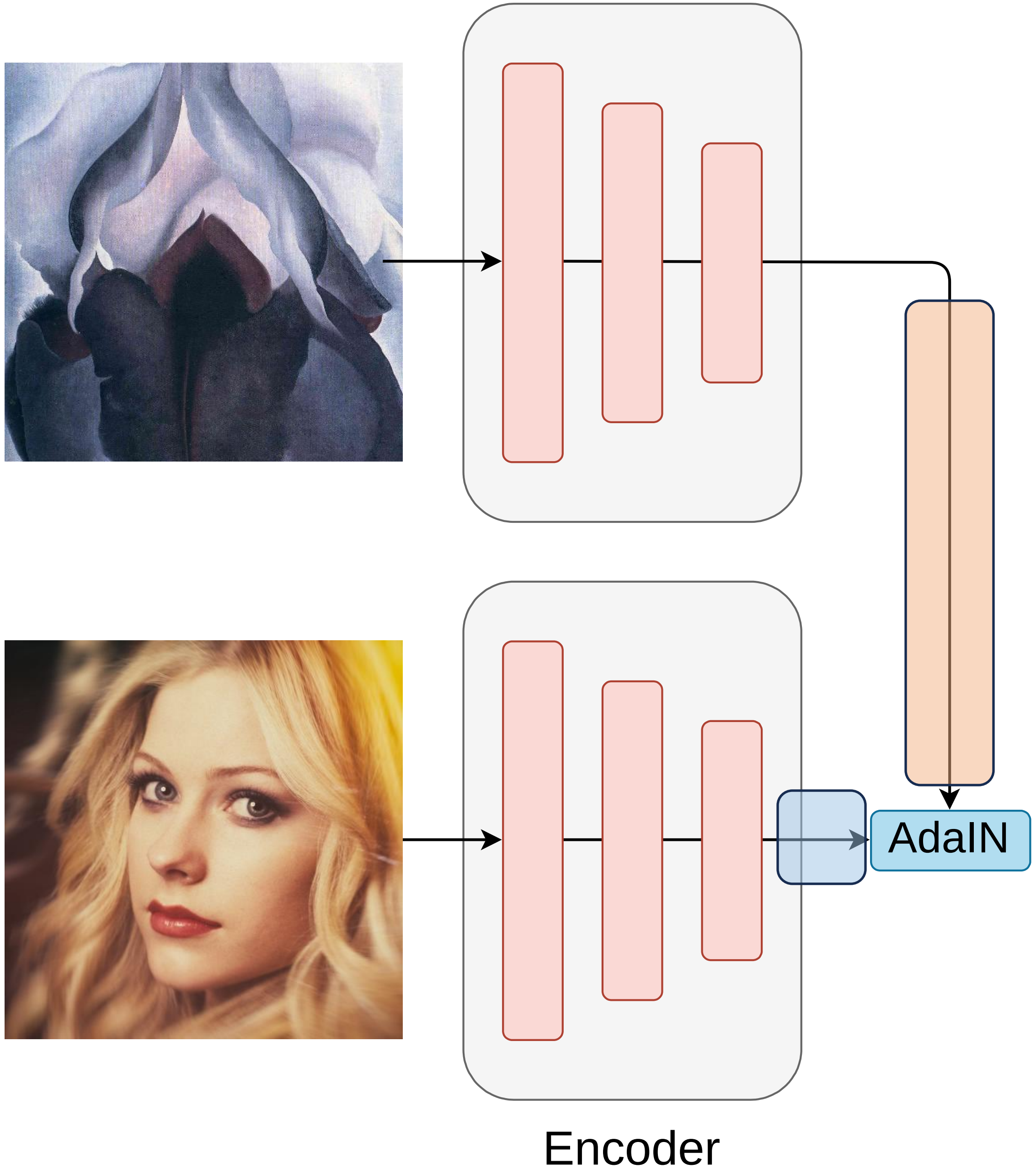
$$\|s_\theta(x; t) - \nabla_x \log p_t(x)\|^2$$

How to condition the network on time steps?

Style Transfer

Style

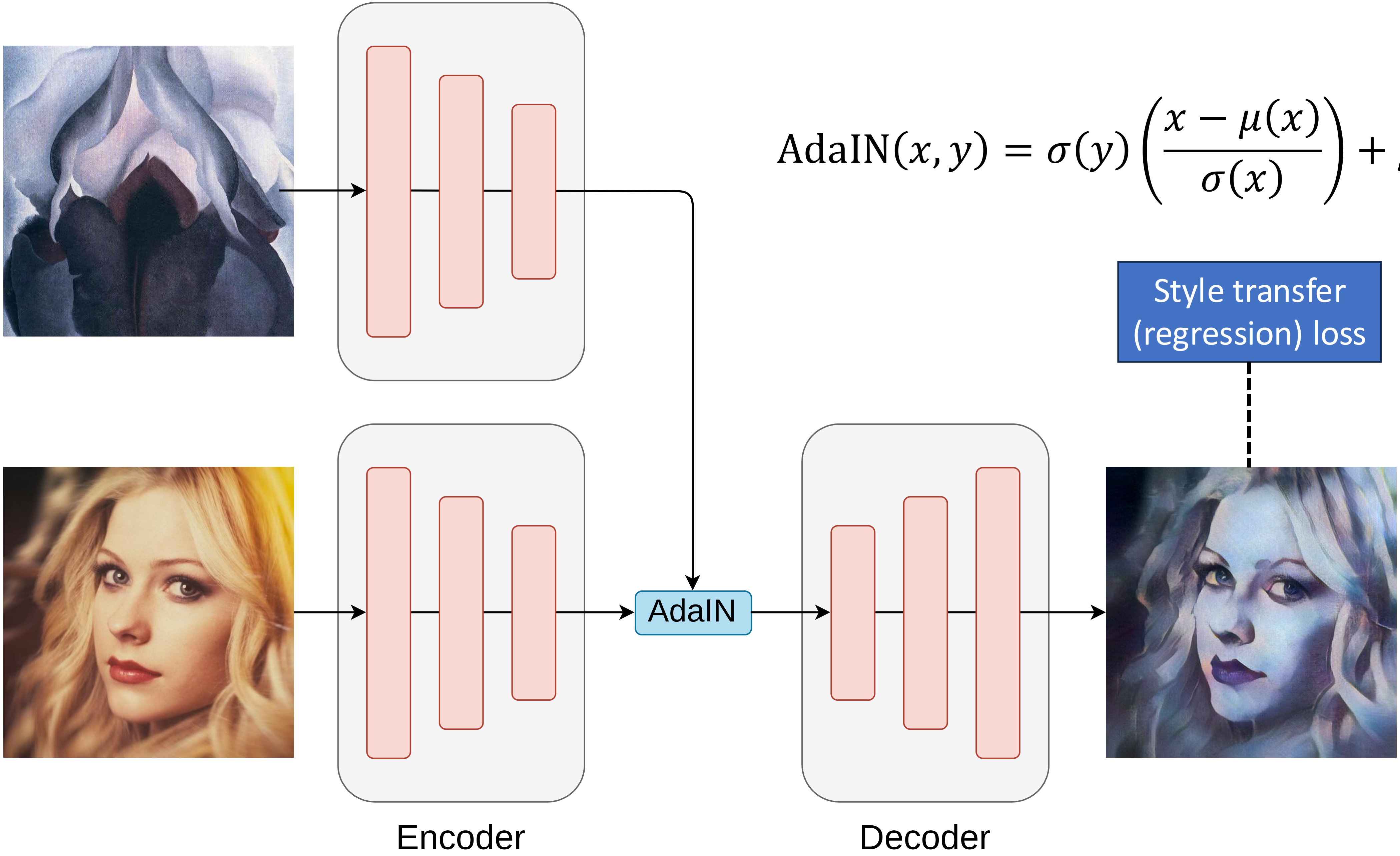
Content



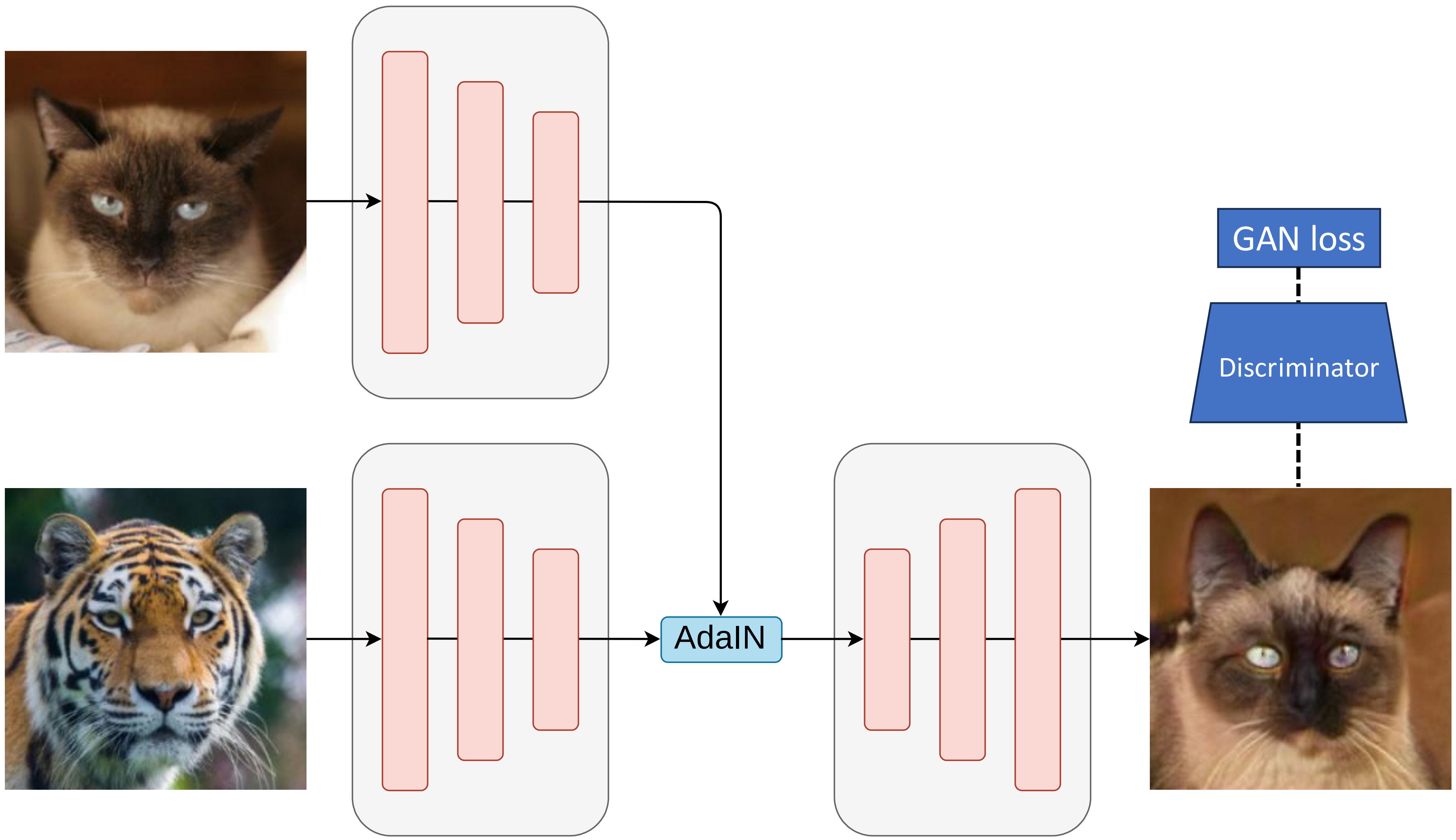
$$\mathrm{AdaIN}(\boxed{x},\boxed{y}) = \boxed{\sigma(y)}\boxed{\left(\frac{x-\mu(x)}{\sigma(x)}\right)} + \boxed{\mu(y)}$$

Huang and Belongie, “Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization”, ICCV 2017

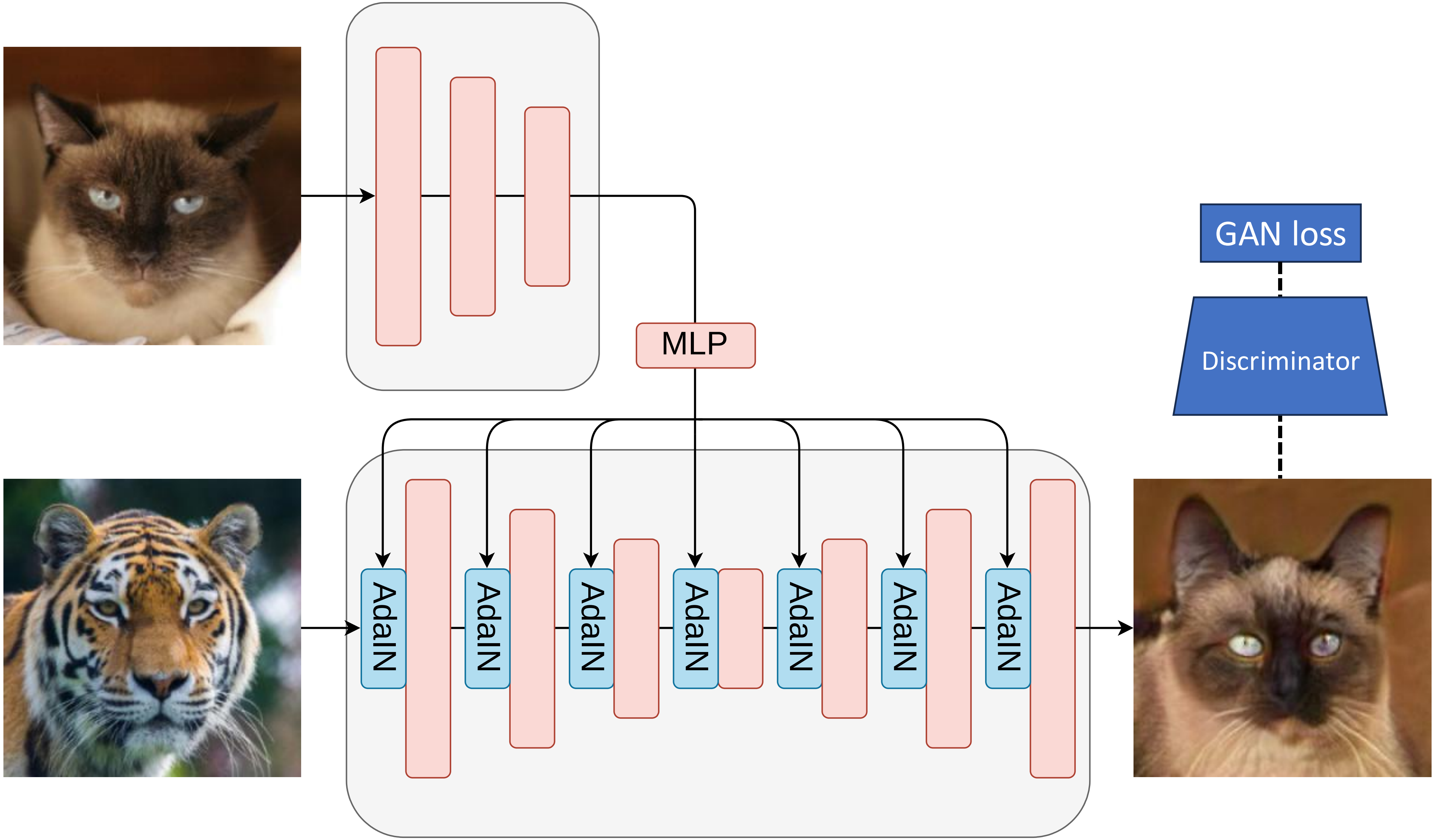
$$\text{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(y)}\right) + \mu(y)$$



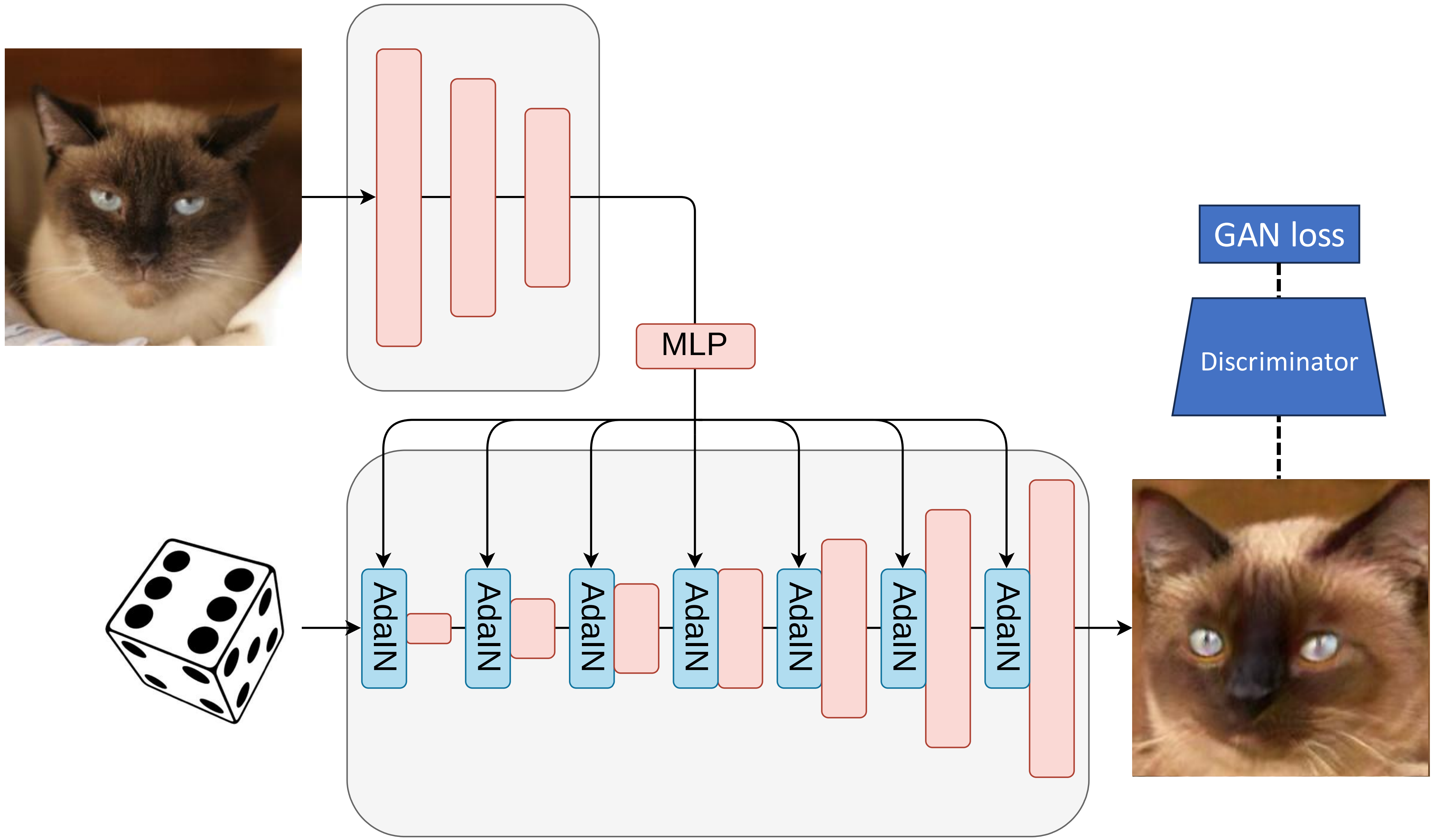
Huang and Belongie, "Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization", ICCV 2017

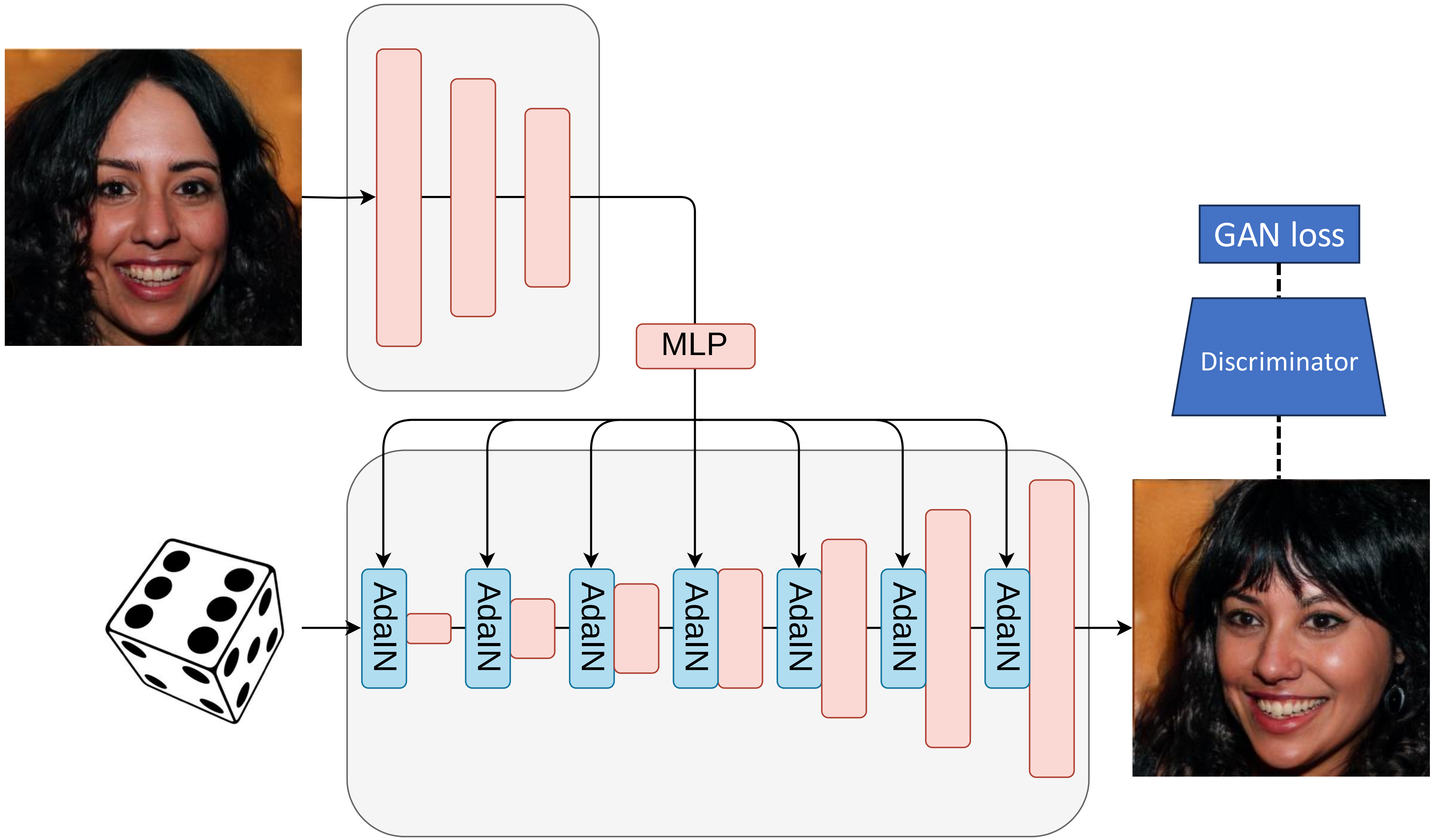


Huang et al., “Multimodal Unsupervised Image-to-Image Translation”, ECCV 2018

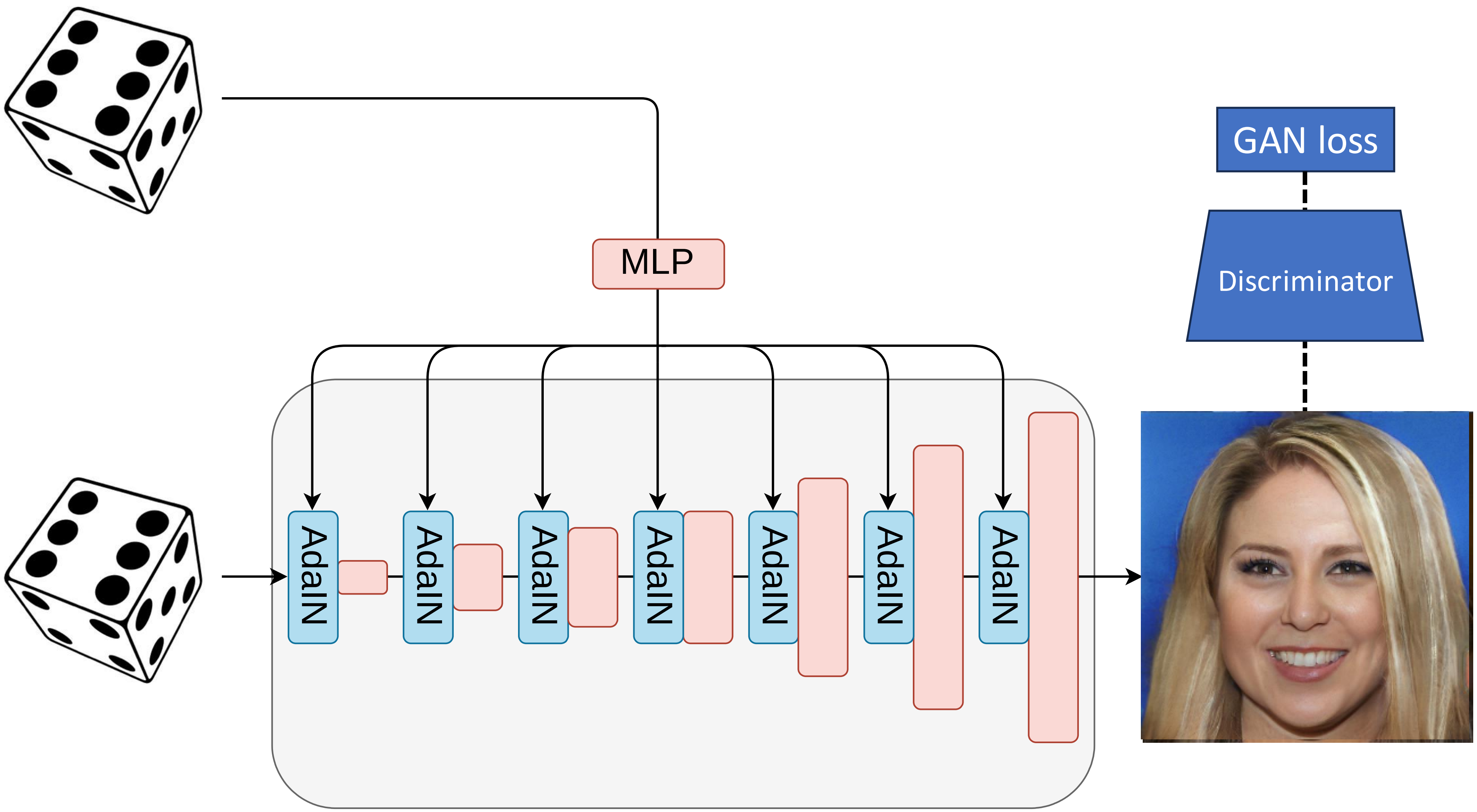


Huang et al., “Multimodal Unsupervised Image-to-Image Translation”, ECCV 2018


MLP
AdaIN
AdaIN
AdaIN
AdaIN
AdaIN
AdaIN
AdaIN
GAN loss
Discriminator



MLP
AdaIN
AdaIN
AdaIN
AdaIN
AdaIN
AdaIN
AdaIN
GAN loss
Discriminator



Karras et al., “A Style-Based Generator Architecture for Generative Adversarial Networks”, CVPR 2019

CNN

These people do not exist. Why websites are churning out fake images of people (and cats)

Thispersondoesnotexist is one of several websites that have popped up in recent weeks using StyleGAN to churn out images of people, cats, anime characters and...

Feb 28, 2019

BBC

AI fake face website launched

A software developer has created a website that generates fake faces, using artificial intelligence (AI).

Feb 19, 2019

VentureBeat

Why ThisPersonDoesNotExist (and its copycats) need to be restricted

A website, launched two weeks ago, that uses Nvidia's publicly available artificial intelligence technology to draw an invented, photo-realistic human being...

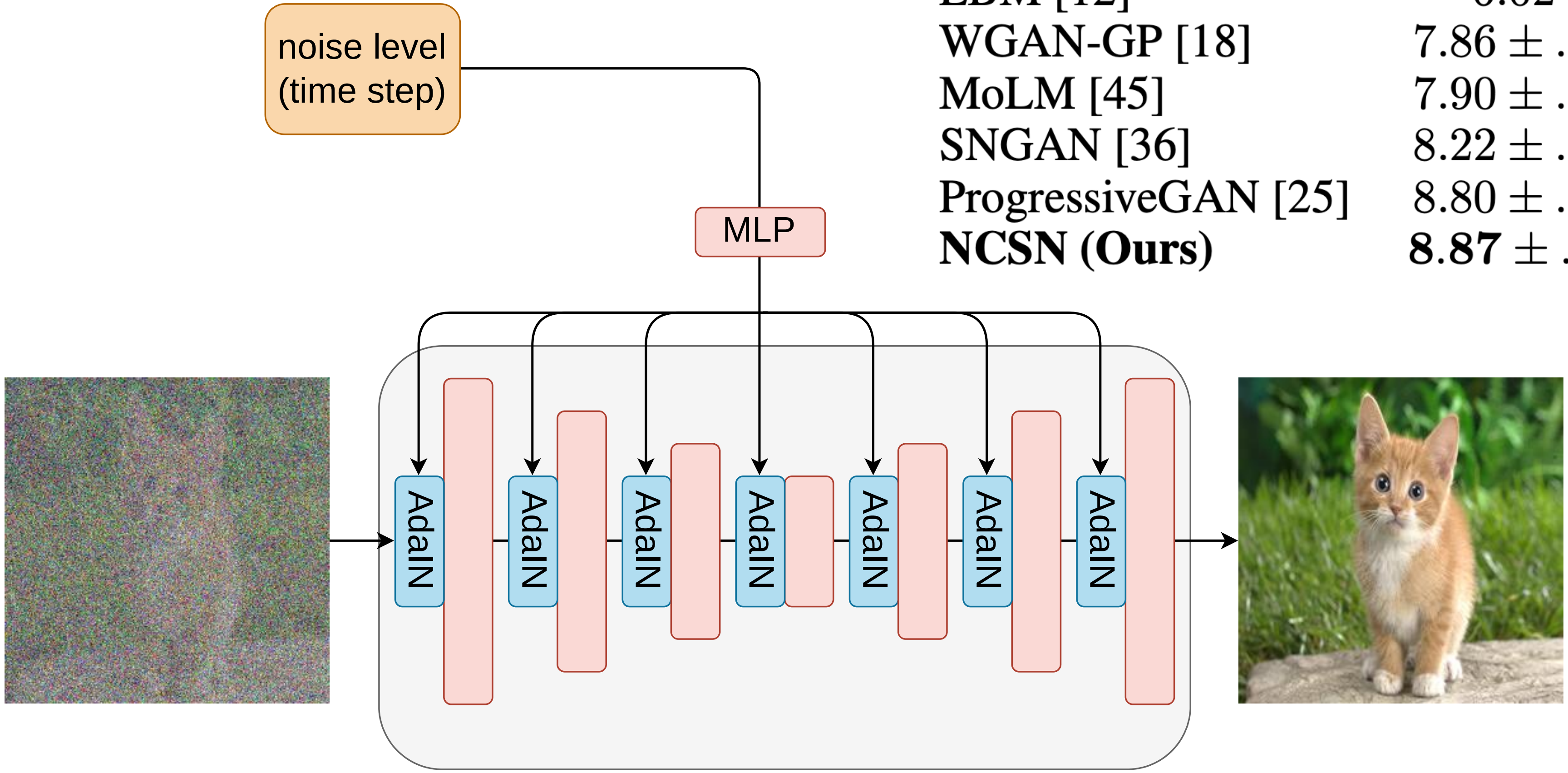
Mar 3, 2019

Noisy as a “style”

Noise level is a **global** property of the image that can be captured by feature statistics!

Noisy as a “style”

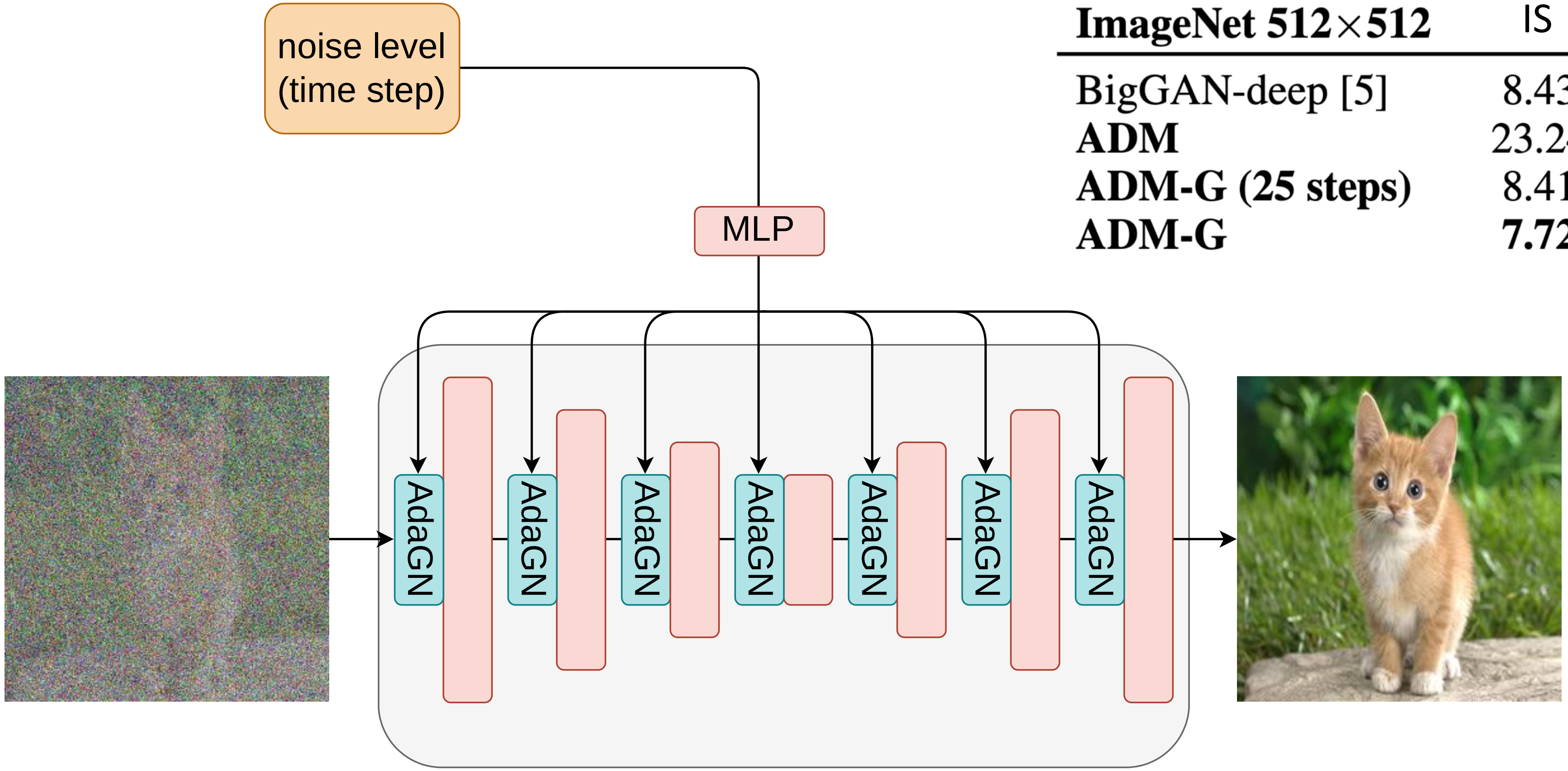
Model	Inception	FID
CIFAR-10 Unconditional		
PixelCNN [59]	4.60	65.93
PixelIQN [42]	5.29	49.46
EBM [12]	6.02	40.58
WGAN-GP [18]	$7.86 \pm .07$	36.4
MoLM [45]	$7.90 \pm .10$	**18.9**
SNGAN [36]	$8.22 \pm .05$	21.7
ProgressiveGAN [25]	$8.80 \pm .05$	-
NCSN (Ours)	$\mathbf{8.87} \pm .12$	25.32



Yang and Ermon, “Generative Modeling by Estimating Gradients of the Data Distribution”, NeurIPS 2019

Adaptive (Instance -> Group) Normalization

“Diffusion models beat GANs” (architecture behind **DALL-E 2**)

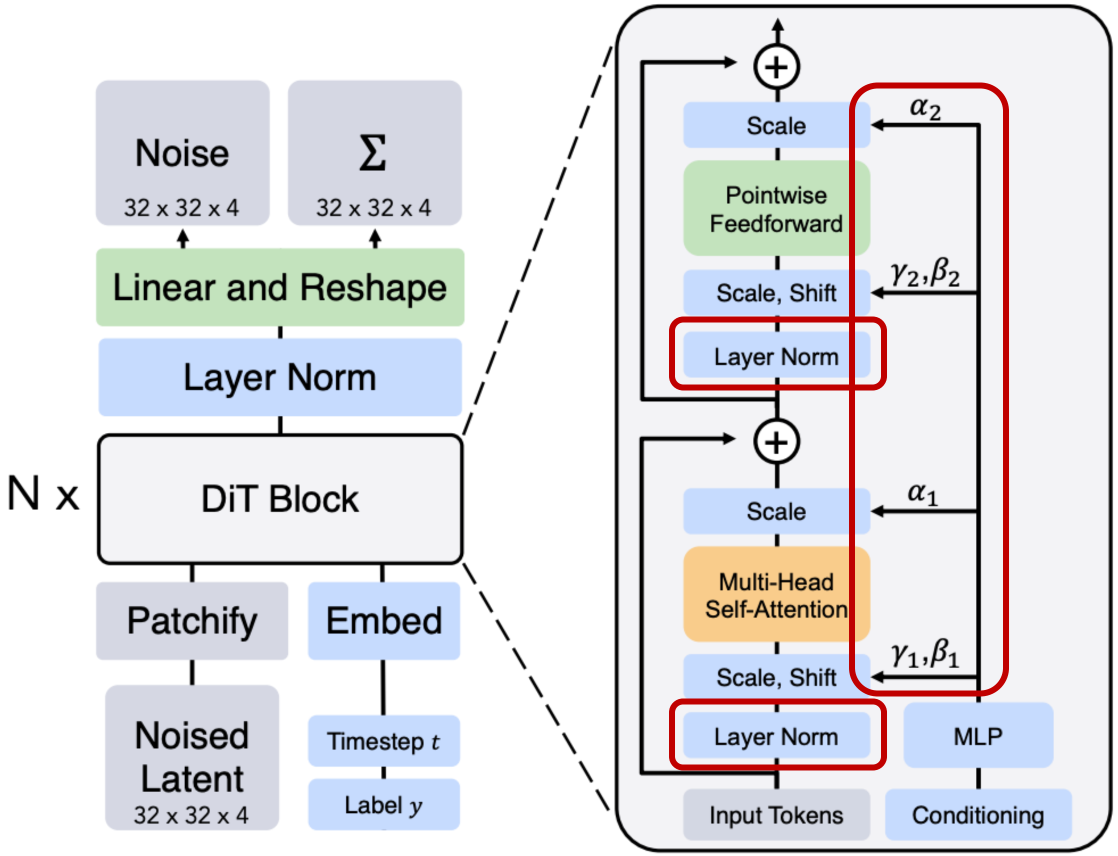


ImageNet 512×512	IS	FID
BigGAN-deep [5]	8.43	8.13
ADM	23.24	10.19
ADM-G (25 steps)	8.41	9.67
ADM-G	**7.72**	**6.57**

Dhariwal and Nichol, “Diffusion Models Beat GANs on Image Synthesis”, NeurIPS 2021

Adaptive (Group -> Layer) Normalization

“Diffusion Transformers” (architecture behind **Sora**)



Peebles and Xie, “Scalable Diffusion Models with Transformers”, CVPR 2023

Directly generating high-resolution images with diffusion is very costly!

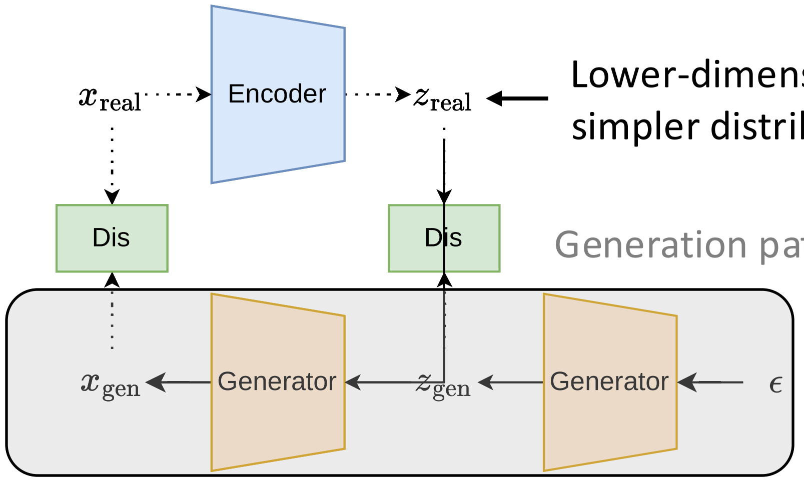
4096*4096 = 17M pixels

Image generated by SANA

Multi-Stage Generative Models

- Divide-and-Conquer
  - Modeling $p(x)$ is hard/expensive
  - $p(x) = \int_z p_\phi(z) p_\theta(x|z)$ and then learn $\boxed{p_\phi(z)}$ and $p_\theta(x|z)$ with two generative models

A non-trivial distribution rather than simple Gaussian!



Lower-dimensional, simpler distribution

Generation path

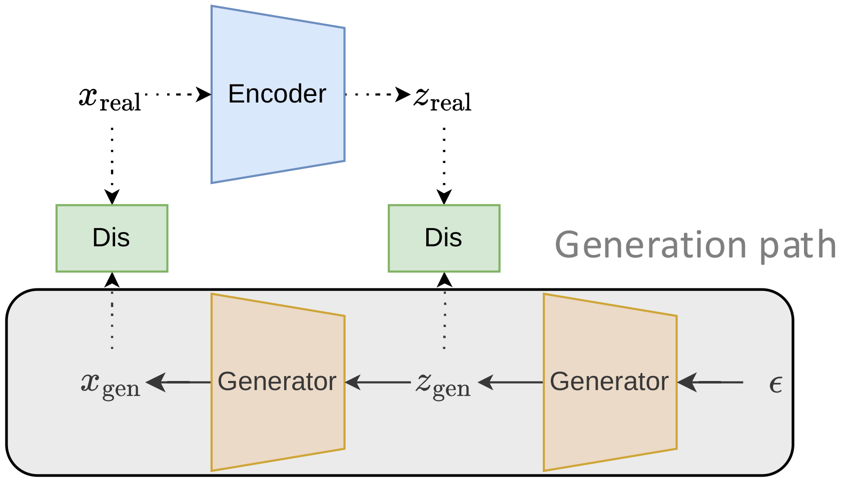
Stacked GANs

Baseline (DCGANs)

Huang et al., “Stacked Generative Adversarial Networks”, CVPR 2017

Multi-Stage Generative Models

- Divide-and-Conquer
  - Modeling $p(x)$ is hard/expensive
  - $p(x) = \int_z p_\phi(z) p_\theta(x|z)$ and then learn $p_\phi(z)$ and $p_\theta(x|z)$ with two generative models

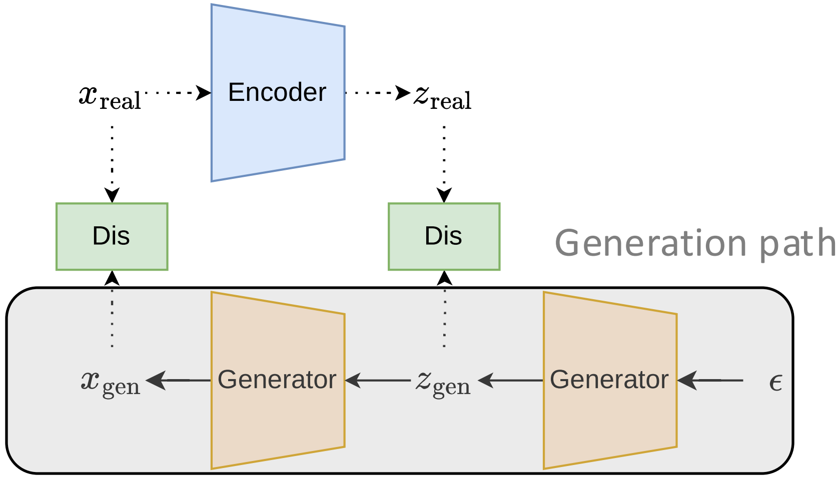


Trained on 1 “Titan X”
(the GPU before 1080)

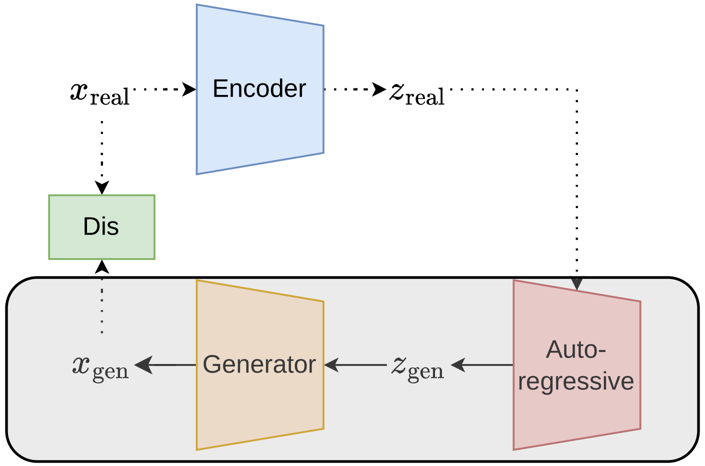
Huang et al., “Stacked Generative Adversarial Networks”, CVPR 2017

Multi-Stage Generative Models

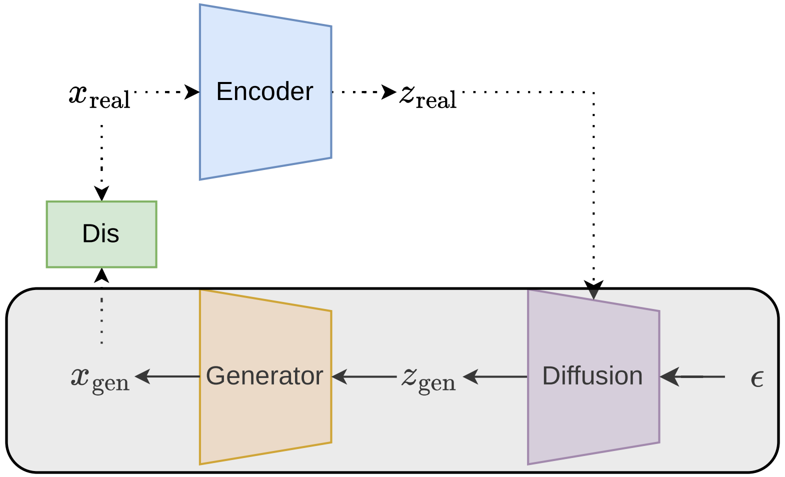
- Divide-and-Conquer
  - Modeling $p(x)$ is hard/expensive
  - $p(x) = \int_z p_\phi(z) p_\theta(x|z)$ and then learn $p_\phi(z)$ and $p_\theta(x|z)$ with two generative models



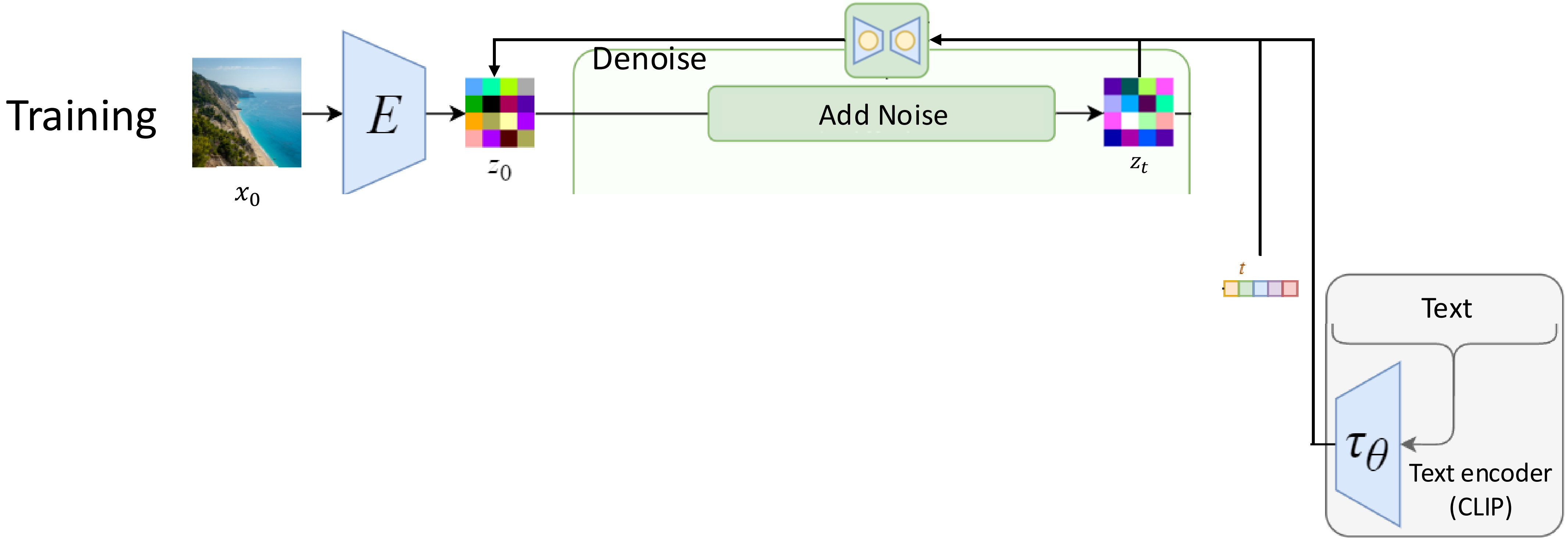
Stacked GANs
Huang et al., CVPR 2017
Image GAN + Latent GAN



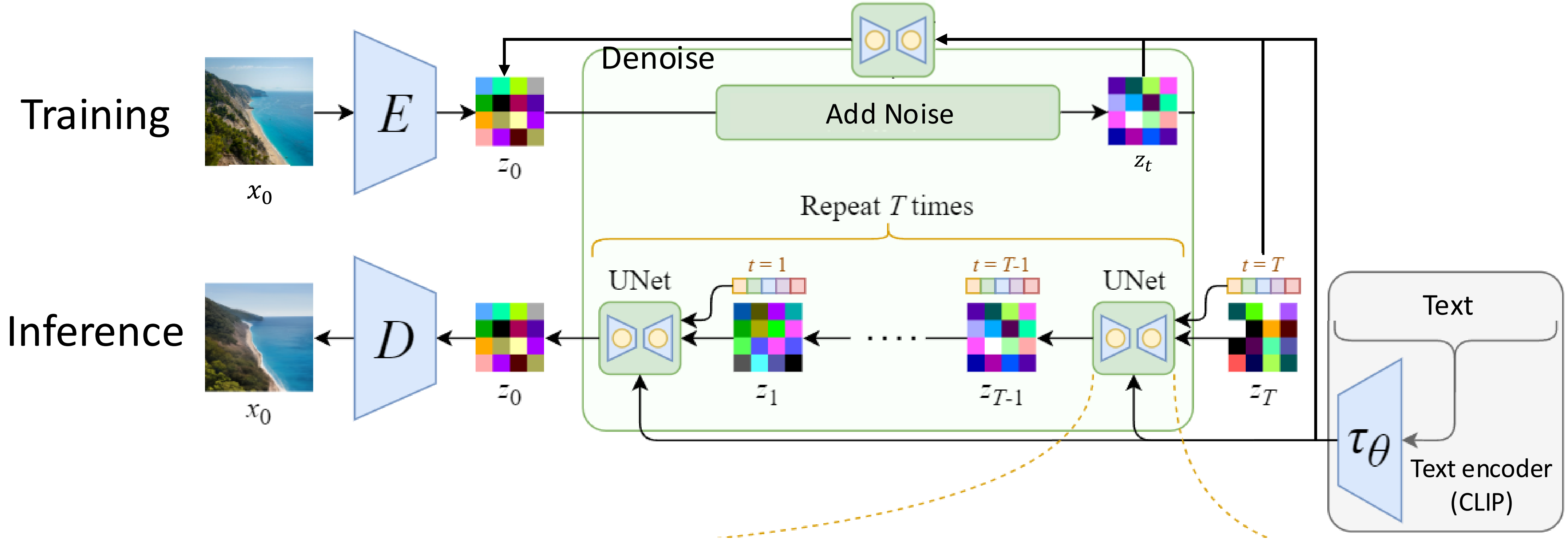
VQGAN
Esser et al., CVPR 2020
Image GAN + Latent AR



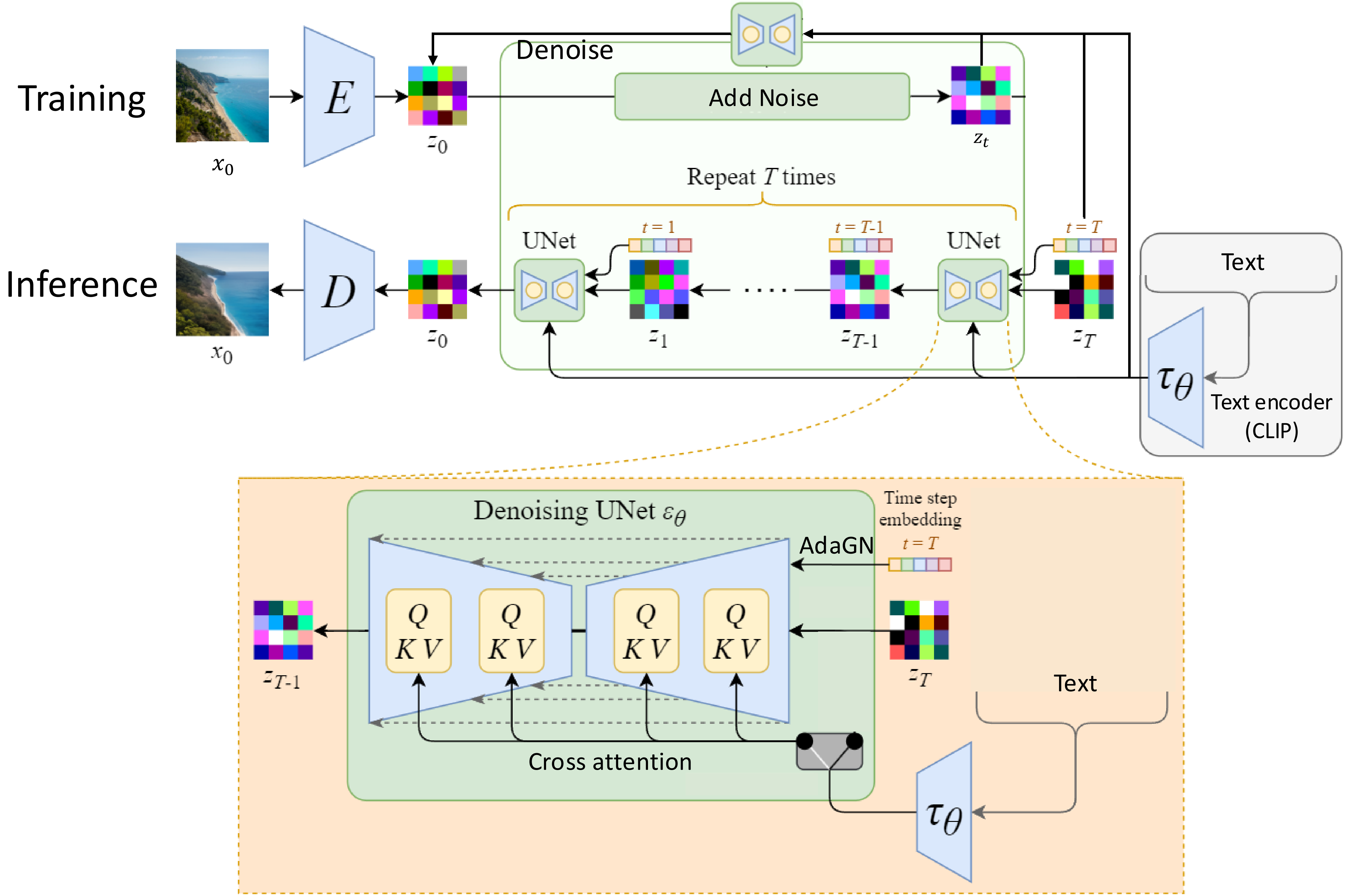
Latent Diffusion Models
Rombach et al., CVPR 2021 **(Stable Diffusion)**
Image GAN + Latent Diffusion



Conditional diffusion: given pairs of (text, image), denoise the image *conditioned* on text

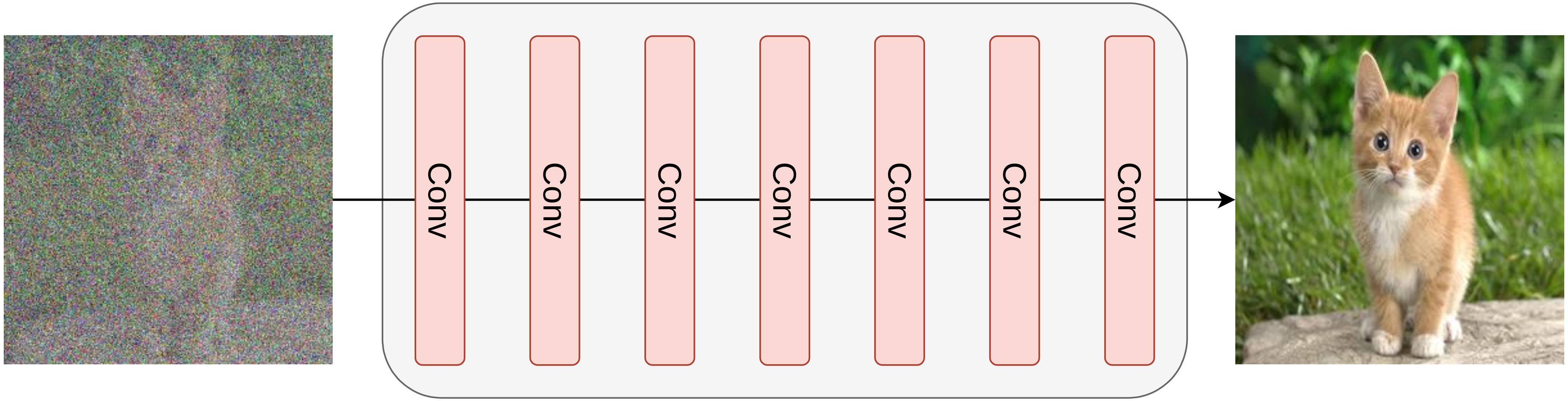


Training
Inference
E
D
x_0
z_0
z_t
Denoise
Add Noise
Repeat T times
UNet
UNet
$t = 1$
$t = T\text{-}1$
$t = T$
z_1
$z_{T\text{-}1}$
z_T
Text
τ_θ
Text encoder
(CLIP)


Training
Inference
Denoise
Add Noise
x_0
E
z_0
z_t
Repeat T times
UNet
UNet
$t = 1$
$t = T\text{-}1$
$t = T$
D
z_1
$z_{T\text{-}1}$
z_T
Text
τ_θ
Text encoder (CLIP)
Denoising UNet ε_θ
Time step embedding
AdaGN
Q K V
Cross attention
Text

Diffusion with naïve CNN architecture

- Limited receptive field
- Computationally expensive

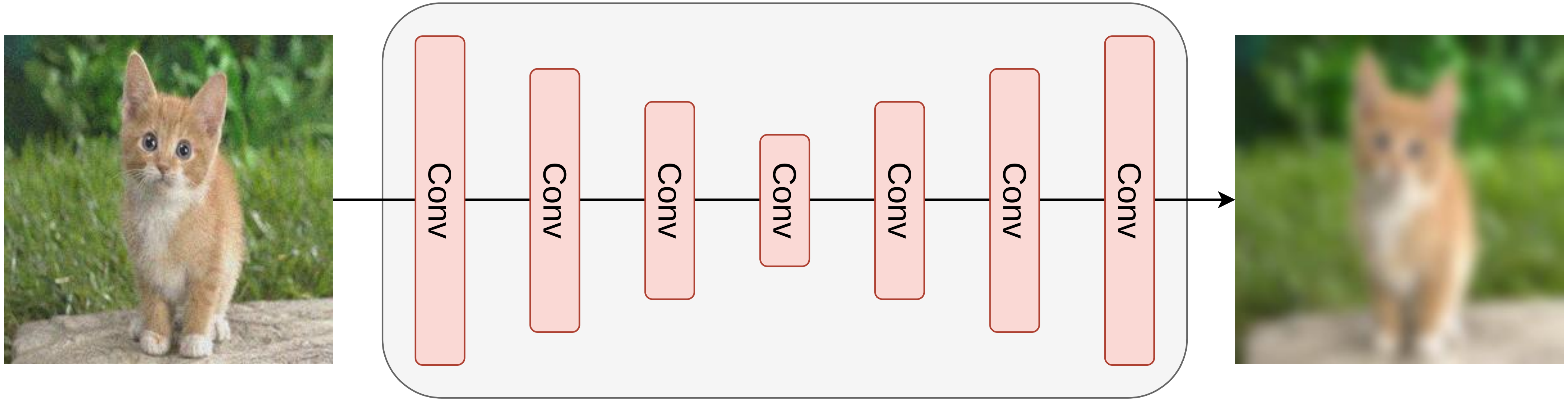


Diffusion with Encoder-Decoder architecture

+ Larger receptive field

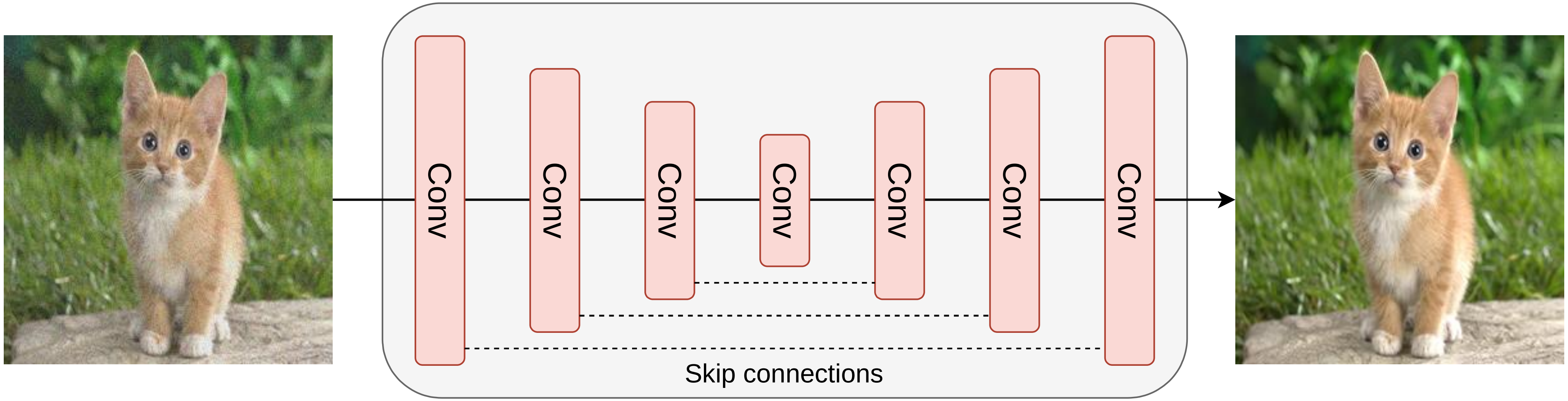
+ Computationally cheap

– Losing details



Diffusion with U-Net architecture

+ Larger receptive field
+ Computationally cheap
+ Preserving details

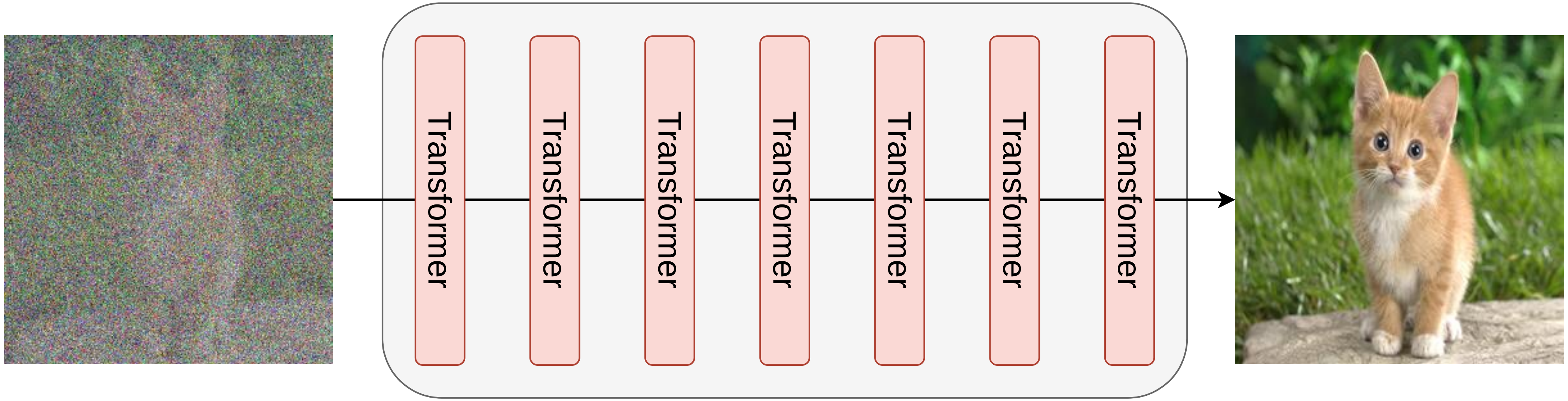


Diffusion with Transformer architecture

+ Full receptive field

– Computationally expensive

+ Preserving details



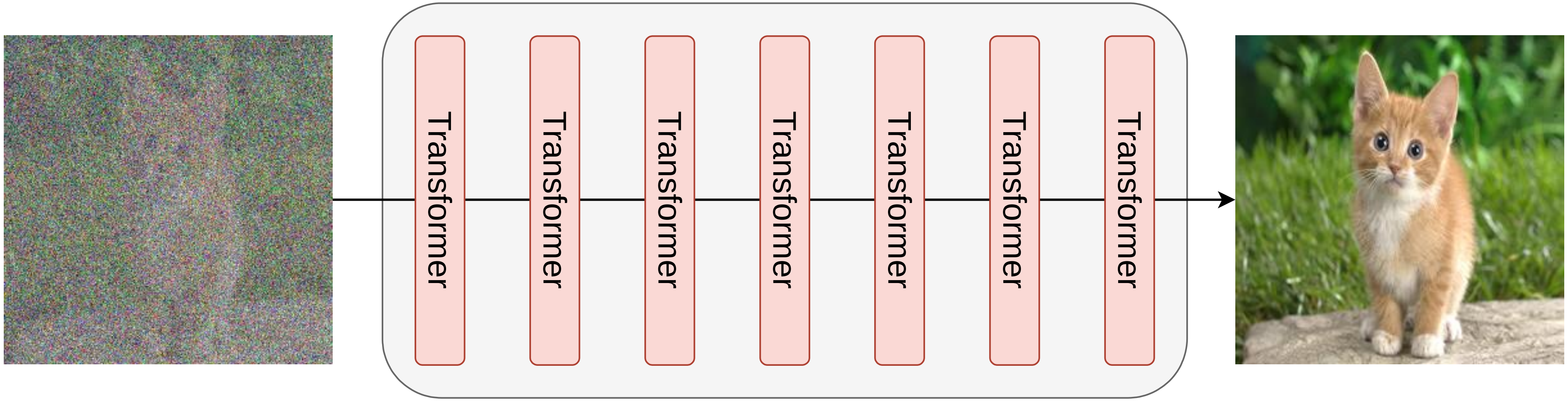
Diffusion with Transformer architecture

(with patchification, efficient implementation, etc.)

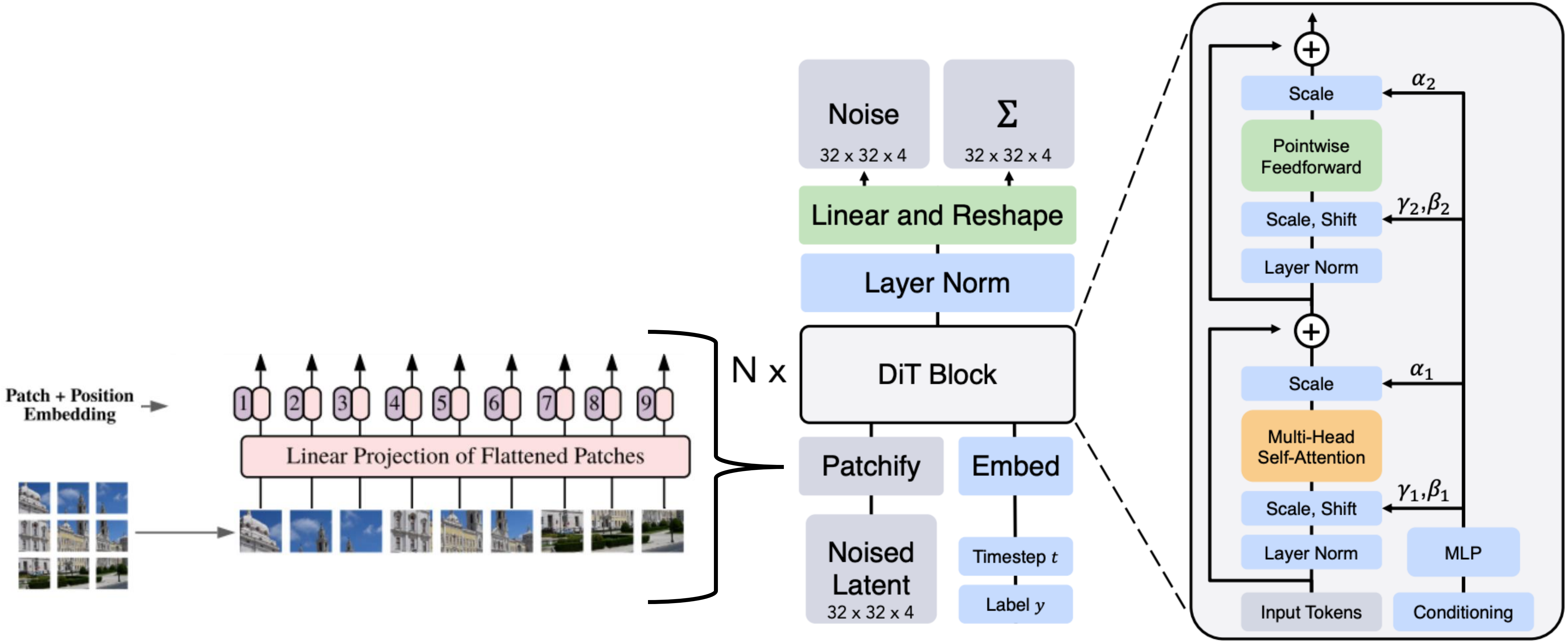
\+ Full receptive field

\+ Computationally feasible

\+ Preserving details

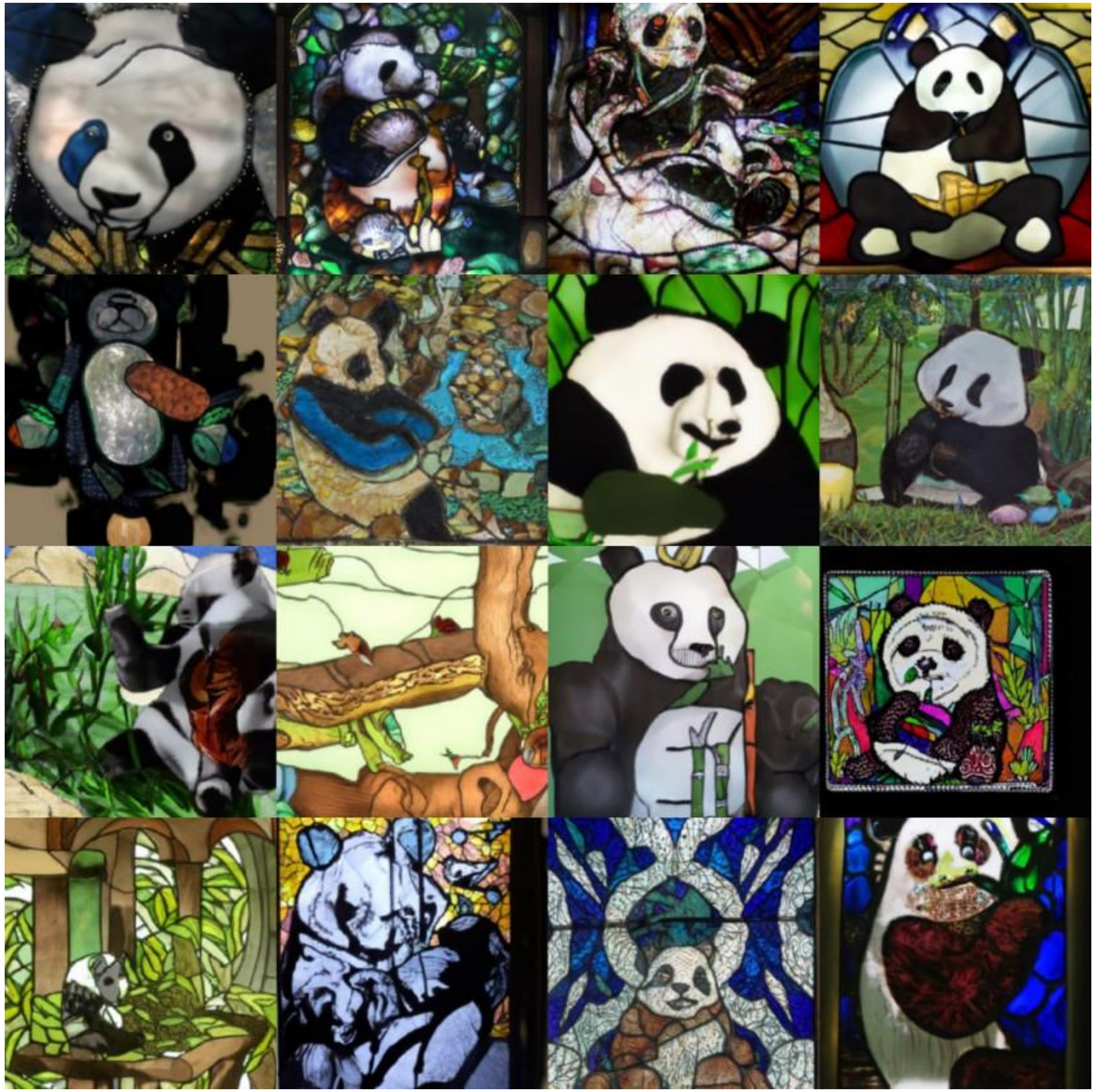


Diffusion Transformer



Outline: Text-to-Image Diffusion Models

- Network architecture
  - Noise-level Conditioning
  - Latent-space Modeling
  - U-Net vs. Transformers
- Classifier-free Guidance (CFG)



Without CFG

With CFG

A stained glass window of a panda eating bamboo.

Results from GLIDE

Classifier-free Guidance (CFG)

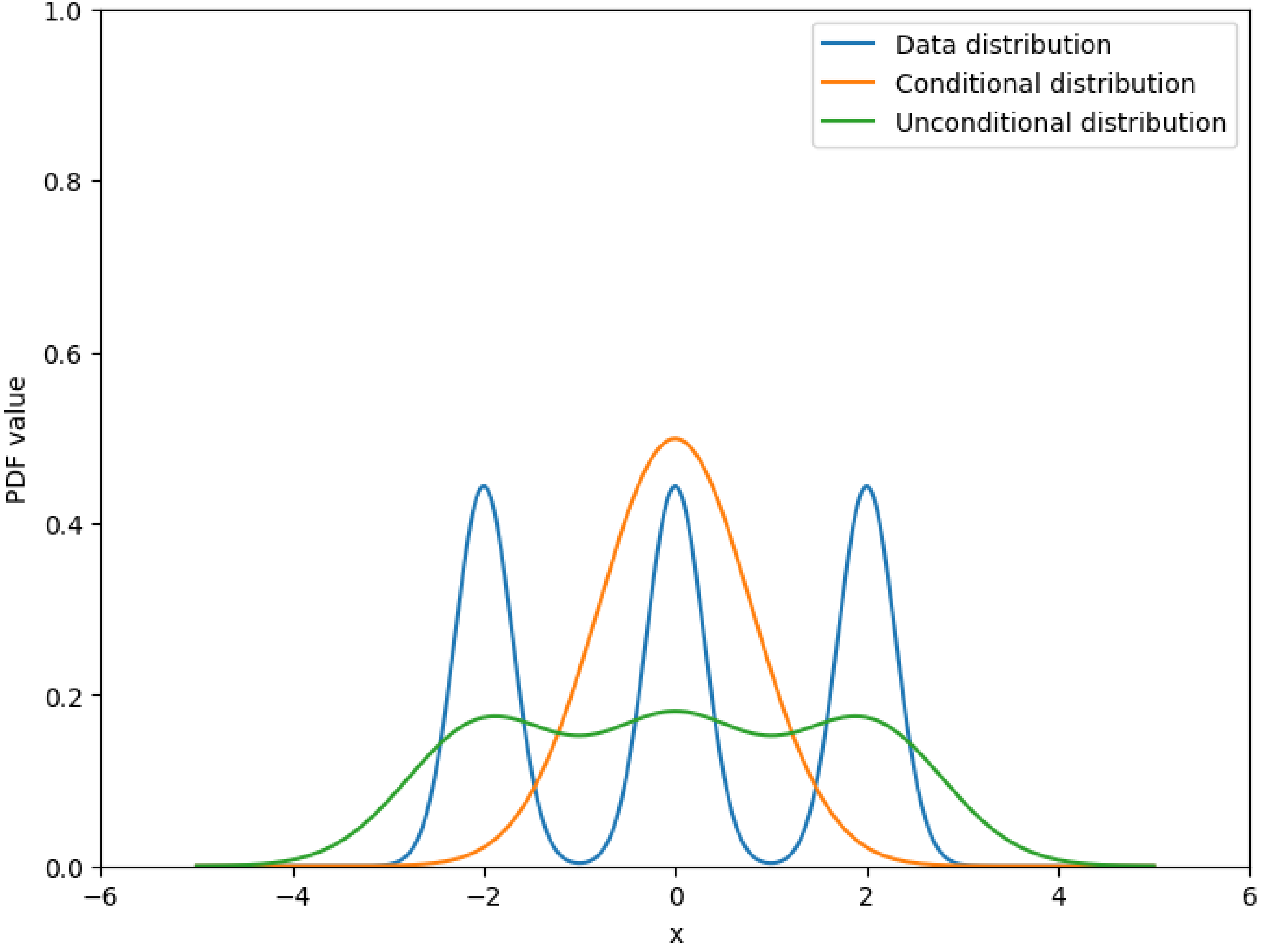
- Drop out the conditioning y (e.g., text) some percentage of the time
- So that the same model learns both
  - Conditional score $\nabla_{x_t} \log p(x_t|y)$
  - Unconditional score $\nabla_{x_t} \log p(x_t)$
- At inference time
  - Instead of using $\nabla_{x_t} \log p(x_t|y)$
  - We use $\nabla_{x_t} \log p(x_t) + \gamma \cdot \left(\nabla_{x_t} \log p(x_t|y) - \nabla_{x_t} \log p(x_t)\right)$ $(\gamma > 1)$

Why does we need CFG?

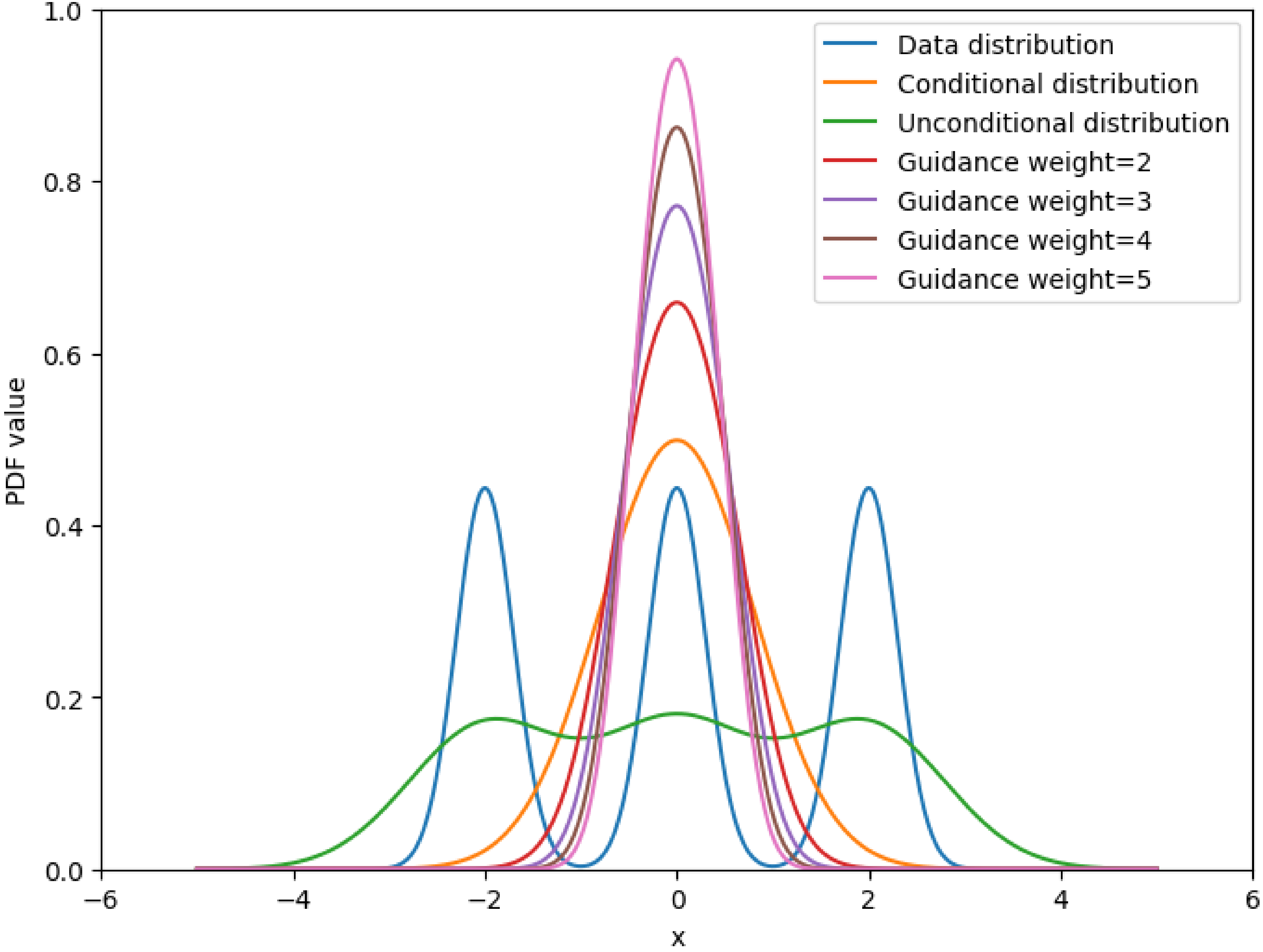
- The diffusion objective learns overly “conservative” distributions
- During training, we sample from our dataset and compute loss
  - If a datapoint is not captured by our model -> huge loss
  - If model distribution contains regions not in data distribution -> probably okay, never saw them during training anyway
  - **Favor recall over precision**
  - **The model distribution is too “flat”, we want to make it sharper**

$$\nabla_{x_t} \log p(x_t) + \gamma \cdot \left(\nabla_{x_t} \log p(x_t|y) - \nabla_{x_t} \log p(x_t)\right) = \nabla_{x_t} \log \left(p_{\text{cond}} \left(\frac{p_{\text{cond}}}{p_{\text{uncond}}}\right)^{\gamma-1} \right)$$

Make the distribution even “more conditional”



Data distribution
Conditional distribution
Unconditional distribution
PDF value
x
1.0
0.8
0.6
0.4
0.2
0.0
−6
−4
−2
0
2
4
6



$$p_{\text{cond}}\left(\frac{p_{\text{cond}}}{p_{\text{uncond}}}\right)^{\gamma-1}$$

5 Minute Quiz

- On Canvas
- Passcode: yoshi